Greedy Geographic Routing in Large-Scale Sensor Networks: A Minimum Network Decomposition Approach

Anne-Marie Kermarrec and Guang Tan*

Abstract

In geographic (or geometric) routing, messages are expected to route in a *greedy* manner: the current node always forwards a message to its neighbor node that is closest to the destination. Despite its simplicity and general efficiency, this strategy alone does not guarantee delivery due to the existence of local minima (or dead ends). Overcoming local minima requires nodes to maintain extra non-local state or to use auxiliary mechanisms. We study how to facilitate greedy forwarding by using a minimum amount of such non-local state in topologically complex networks. Specifically, we investigate the problem of decomposing a given network into a minimum number of *Greedily Routable Components* (GRC), where greedy routing is guaranteed to work. We approach it by considering an approximate version in a continuous domain, with a central concept called the *Greedily Routable Region* (GRR). A full characterization of GRR is given concerning its geometric properties and routing capability. We then develop simple approximate algorithms for the problem. These results lead to a practical routing protocol that has a routing stretch below 7 in a continuous domain, and close to 1 in several realistic network settings.

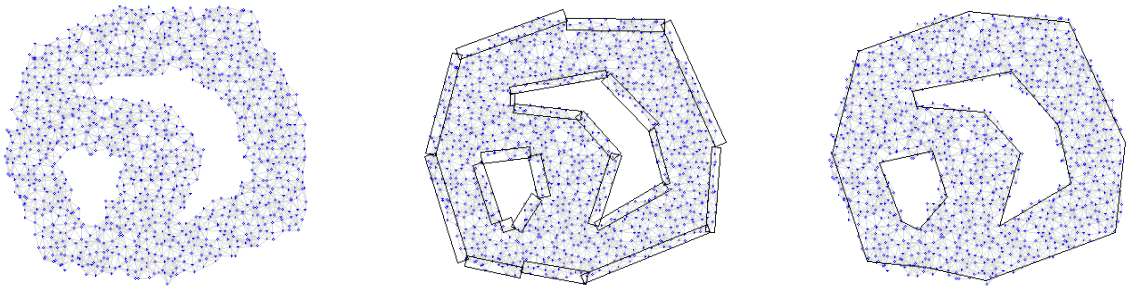
1 Introduction

Geographic routing has been shown to be a promising method for efficient point-to-point routing in large-scale wireless sensor/ad hoc networks. In this method, it is assumed that every node knows its own location in the plane, and the source of a message knows the location of the destination through a location service system. The message is expected to proceed in a greedy manner – it is always forwarded to a node that is closest to the destination among the forwarder's neighbors. Such a greedy strategy, were it to succeed, can often produce a low stretch path [12]. One of the major advantages of such a method is that it is *low-state*, in that every node only needs to remember the location information of its immediate neighbors, thus fitting in the resource-constrained environment of a sensor network.

Unfortunately, greedy routing alone does not guarantee successful delivery of messages in a practical network, due to the existence of *local minima*, where a node does not have a neighbor closer than itself to the destination. This problem can be solved by face (or perimeter) routing [12] or expanding ring search [22] (i.e., incrementally scoped flooding), possibly at the cost of significantly increased stretch [13] or excessive message transmission [23]. To address these shortcomings, researchers have proposed to exploit non-local topological information, using some global data structures such as landmark Voronoi complex [8], medial axis graph [5], and visibility graph [24], to find low-stretch routes. An alternative approach is based on the idea of divide and conquer: the network is decomposed into components [8, 23, 29] where greedy routing is likely to perform well, and then a global structure is used to assist inter-component routing.

In this paper we focus on the decomposition approach. To improve on previous work which concentrates on feasibility, our goal is to design a low-stretch routing protocol that uses a minimum number of network components. This is important for every node to keep a small routing table since the number of components

*Contact author.



(a) Original network topology (b) Rectangles for smoothing boundaries (c) The profile polygons

Figure 1: Profile polygons for a large-scale network.

directly determines the amount of non-local state per node. We ask: *Given a network, how to decompose it into a minimum number of components such that in each component, greedy geographic routing alone guarantees delivery for every pair of nodes?*

In this paper, a network that permits every-pair purely greedy routing is called a *Greedily Routable Component* (GRC). A real-world network may contain a large number of local minimum points, thus may generate many components, rendering the solution unattractive. Our strategy is to focus on the major performance factors by considering the problem in a continuous domain. Given a network deployed on a plane, we extract the most significant geometric features (e.g., large holes) of the field, by creating *profile polygons* of the network; see Figure 1 for an example. A profile polygon is simply a smoothed boundary polygon of a network, generated at a base station using the sensor field's geometric map when it is available, or using a protocol described in [24] when the map is unavailable.[1] We then consider how to decompose such a polygonal area into a minimum set of *Greedily Routable Regions* (GRRs) that permit every-pair purely greedy routing on the plane.

The continuous version of the network allows us to concentrate on the field's high order geometric properties. It is exactly those features, arising from irregular terrains, task or security boundaries, etc, that dominate the performance of greedy routing [5, 8, 24]. Such features may form "traps" that mislead a message far away in a wrong direction, resulting in a big detour. Our model attempts to capture those main causes for the performance degradation. In contrast to the global features, local connectivity irregularity only has a relatively small impact on path quality. Practical sensor networks are required to maintain a certain degree of connectivity and sensing coverage [4, 26] for service dependability reasons, so the node distribution can be seen as approximately uniform in a local scope. It follows that local minima in such a scope can be overcome with simple strategies at a low cost. For example, with the widely used grid-guided [5, 24, 27, 28] or uniformly random distribution [5, 12, 26], a node may route out of a local minimum by searching its neighborhood in only a few hops.

We establish necessary and sufficient conditions for a polygonal area to be a GRR (Section 2). An important finding is that a single GRR may minimally contain an arbitrarily large number of convex components. This implies that, with respect to keeping low global state, the convex decomposition approach suggested in [23] can perform arbitrarily worse than the GRR approach. Based on the properties revealed, we further prove that greedy routing in a GRR has a stretch lower than 3.

We next show that the minimum GRR decomposition problem is NP-hard, which motivates us to design a simple, albeit sub-optimal, algorithm (Section 3). Graph embedding techniques are also considered as an optimization (Section 4). The algorithm is used to design a practical routing protocol (Section 5) that has a stretch below 7 in a continuous domain, and close to 1 in several realistic experimental settings (Section 6).

[1]The protocol, working on a planarized network, identifies *big faces* whose sizes are greater than a threshold value, and then determines a profile polygon for each of them.

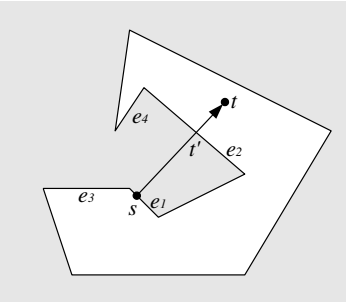


Figure 2: Conflict Relationship. e_1 and e_2 conflict with each other; e_3 conflicts with e_4 but e_4 does not conflict with e_3.

We give a brief review of related work in Section 7, before we conclude the paper in Section 8.

2 The minimum GRR decomposition problem (Min-GRR)

In this section we establish notation and precisely define the problems. A *polygonal environment* $\mathcal{P}$ is a set of points on the plane enclosed by a set of boundaries $P_0, P_1, \ldots, P_k$, where P_0 is the *outer boundary* and $P_{i>0}$ are boundaries of *holes* of $\mathcal{P}$. Each boundary P_i is a simple polygon (whose non-adjacent edges do not intersect) and consists of an ordered set of polygonal vertices, which defines a set of edges. $\overline{\mathcal{P}}$ is called $\mathcal{P}$'s *external region*. For a polygonal vertex u, its host polygon is denoted $P(u)$. The *inner angle* of u, denoted $\angle u$, is defined as u's polygonal angle that spans across its neighborhood in $\mathcal{P}$. u is called a *notch vertex* if $\angle u > \pi$. The number of notch vertices of $\mathcal{P}$ is denoted $n(\mathcal{P})$. The number of boundary polygons of $\mathcal{P}$ is denoted $b(\mathcal{P})$.

A simple polygonal area C is a component of $\mathcal{P}$ if $C \subset \mathcal{P}$. A set of components $\{C_i\}$ is a *decomposition* of a polygonal environment $\mathcal{P}$ if their union is $\mathcal{P}$ and all C_i are interior disjoint. Let $|pq|$ denote the Euclidean distance between two points p and q. A path between two points s and t is also called an st-path. Let $D_g(s,t)$ denote the Euclidean length of an st-path produced by a greedy routing algorithm, and $D_{\min}(s,t)$ denote the *geodesic* distance (shortest path distance) between s and t inside $\mathcal{P}$.

In a given polygonal environment $\mathcal{P}$, a *routing hop* corresponds to a non-degenerate straight line segment that lies entirely in $\mathcal{P}$. Loosely speaking, successful greedy routing requires that starting from an arbitrary point s in $\mathcal{P}$, the algorithm can always make a routing hop that brings the current point closer to the destination. More formally, we have the following definition.

Definition 1. Greedily Routable Region (GRR). *Let s and t be two arbitrary and distinct points in a polygonal environment $\mathcal{P}$. If s can always make a routing hop within $\mathcal{P}$ to some point s' such that $|s't| < |st|$, then $\mathcal{P}$ is a* Greedily Routable Region.

A *GRR decomposition* of $\mathcal{P}$ is a decomposition $D(\mathcal{P})$ in which all components are GRRs. Our aim is to decompose $\mathcal{P}$ into a minimum set of GRRs. To characterize a GRR, we need the following definition.

Definition 2. Conflict Relationship. *Let e_1 and e_2 be two edges of a polygonal environment $\mathcal{P}$. Define a* perpendicular outward ray *(POR) of e_1 to be a ray such that (1) it starts from a non-end point on e_1, (2) it is perpendicular to e_1, and (3) it crosses the external region of $\mathcal{P}$. If there exists a POR of e_1 that intersects with e_2, then e_1 is said to* conflict with e_2.

Two edges are said to be *in conflict* if one of them conflicts with the other. Figures 2 provides several examples of conflict relationship. Note that conflict relationship is not necessarily symmetric; that is, e_1

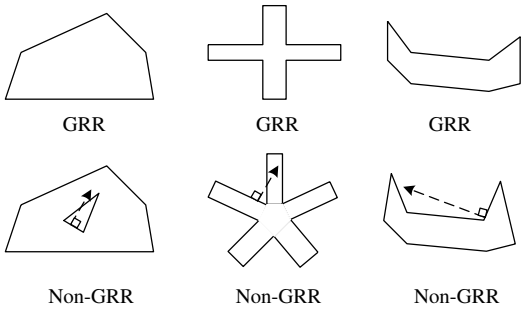


Figure 3: Examples of GRRs and non-GRRs.

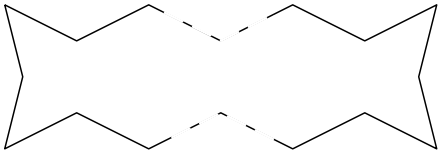


Figure 4: A GRR need not be a convex polygon.

conflicting with e_2 does not imply e_2 conflicting with e_1. The condition for a polygonal environment to be a GRR is as follows.

Theorem 1. *A polygonal environment is a GRR if and only if it has no two conflicting edges.*

Proof. Assume a polygonal environment $\mathcal{P}$ as shown in Figure 2. If $\mathcal{P}$ contains two conflicting edges, say e_1 and e_2, then there must exist a POR from some point s on e_1 that intersects with e_2 at point t'. Clearly, s cannot make a hop within $\mathcal{P}$ that takes itself closer to t', so $\mathcal{P}$ is not a GRR.

If $\mathcal{P}$ is not a GRR, then there exists some point s that cannot make a routing hop to another point t. s must be on some boundary polygon, say in the position shown in Figures 2, then st must be perpendicular to e_1 and across the external region of $\mathcal{P}$, otherwise there exists a point in s's neighborhood within $\mathcal{P}$ that is closer to t. This means that st is a POR of e_1, so e_1 conflicts with e_2. □

Figure 3 provides several examples of GRRs and non-GRRs. From Theorem 1 and Figure 3, it is easy to see that a GRR cannot contain holes, that is, it must be a simple polygon. It is important to note the difference between GRRs and convex polygons: while a convex polygon is obviously a GRR, a GRR does not need to be convex. In fact, a GRR can minimally contain an arbitrary number of convex components, since its inner angle can be greater than π; see Figure 4 for an example.

The performance of a routing algorithm is often measured by *stretch*. The stretch of an st-path is the ratio of the path length to $D_{\min}(s,t)$. A routing algorithm's stretch is defined as the worst-case stretch for all possible source-destination pairs.

Theorem 2. *A greedy routing algorithm in a GRR has a stretch smaller than* 3.

Proof. Consider a source-destination pair s and t in a GRR. It is known [2] that the shortest st-path is a polygonal chain between s and t and whose inner vertices are the polygonal vertices of the GRR; see Figure 5(a) for an illustration. First, we claim that a greedy st-path must pass through all the vertices of this polygonal chain. To see this, suppose that the greedy path begins at a polygonal vertex u on this chain (initially $u = s$). The greedy routing strategy will produce a path in the direction $u \rightarrow t$, until the path hits the GRR boundary at some point u_1, then the path will proceed along the boundary of GRR until reaching a GRR vertex u_2, and then resumes the straight line move toward t, and so on. The angle $\angle u u_1 u_2$ must

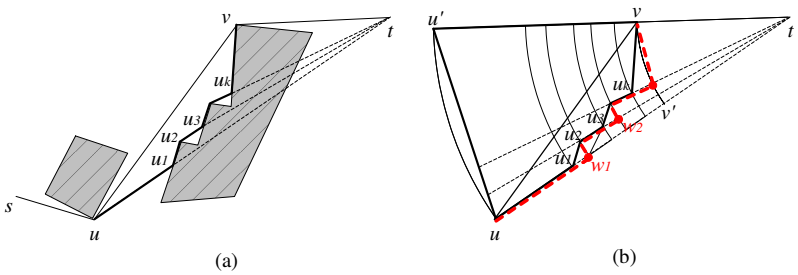


Figure 5: In a GRR, greedy routing between s and t has a stretch below 3.

be greater than $\pi/2$, since otherwise we would be able to find a perpendicular outward ray from u_1 that intersects with the shortest vt path, which implies that this ray will intersect with some boundary edge of the GRR, violating the conditions for a GRR (Theorem 1). Thus the greedy uv-path must be entirely within the triangular area uvt. The behavior of greedy routing itself implies that the greedy uv-path must reach v.

Now take a closer look at the greedy uv-path as shown in Figure 5(b). This path consists of a series of line segments $uu_1, u_1u_2, \ldots, u_kv$ that step progressively closer to the destination point t, as illustrated by the series of co-centric circles. Beside the greedy uv-path shown in bold lines, an artificial path is shown in a dashed (red) polygonal chain, which is at least as long as the greedy uv-path. It is possible to verify that the length of this artificial path is less than $|uu'| + |uv'| = |uu'| + |u'v|$. Since $\angle vu'u = \angle v'uu' > \angle vuu'$, we have $|u'v| < |uv|$, which gives $|uu'| + |u'v| < 2|u'v| + |uv| < 3|uv|$. It follows that the length of the greedy uv-path, $D_g(u, v)$, is less than $3|uv|$. Combining all the greedy paths between s and t, we can conclude that $D_g(s, t) < 3D_{\min}(s, t)$. This proves the theorem. □

Next we establish Min-GRR's NP-hardness by reduction from a modified Planar 3SAT (P-3SAT) problem [14].

Theorem 3. *The Min-GRR problem is NP-hard.*

Proof. We use a modified Planar 3SAT (P-3SAT) problem [14] to show Min-GRR's hardness. The proof is inspired by [18, 21], and is provided in Appendix A. □

3 GRR-Decomp: A Simple GRR Decomposition Algorithm

In this section we describe a simple GRR decomposition algorithm, referred to as *GRR-Decomp*.

GRR-Decomp is a centralized algorithm and is meant to be run on a control point such as a base station. Though it is centralized, this solution is simple and suffices for most application scenarios we can envision. Notice in many of the real-world environments, the network's high order topological features (e.g., big holes) often well reflect the structure of the environment (e.g., physical boundaries and obstacles) and so will be few. It is those features that the algorithm needs to deal with, thus the running time is determined only by the complexity of those features, while is independent of the number of network nodes. For example, in a campus or factory environment, GRR-Decomp's running time will largely depend on the number and layout of buildings, which is often small, although the number of deployed sensors could be many. Moreover, the environmental structures usually remain relatively stable, so topological changes do not happen frequently at a large scale. This means that the centralized planning only needs to be done sparingly in order to adapt to possible dynamics due to, for example, depletion of energy or external damages. (Throughout the paper we assume a static sensor network.)

The algorithm begins with the assumption that the profile polygons (see Figure 1) of a network have already been determined and reported to the base station. The polygonal environment is denoted $\mathcal{P}$ and

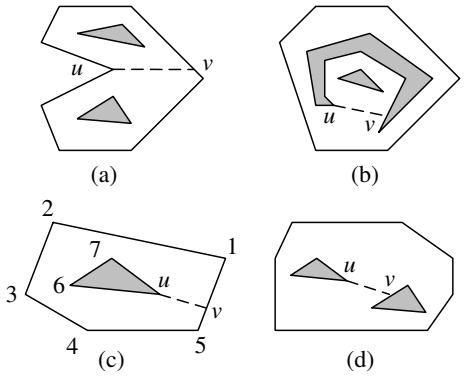


Figure 6: Illustration of GRR decomposition. (a,b) a bisector splitting the current polygonal environment; (c,d) a bisector modifying the current polygonal environment.

Algorithm 1: **Conflict-Resolve($\mathcal{P}$)**

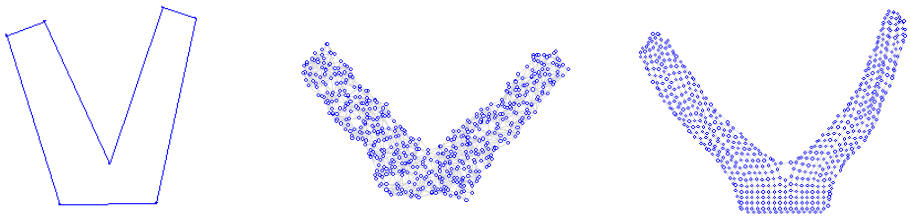
input : A polygonal environment $\mathcal{P}$
output: A decomposition of $\mathcal{P}$, $\{C_i\}$, such that every C_i is a GRR.

while $\mathcal{P}$ *contains a non-GRR polygon* P_i **do**
  Let p be a vertex of P_i that has maximum inner angle;
  Draw a bisector of p's inner angle, yielding one or two new polygonal environments $\{\mathcal{P}_i\}$;
  foreach $\mathcal{P} \in \{\mathcal{P}_i\}$ **do Conflict-Resolve($\mathcal{P}$)**;
end
if $\mathcal{P}$ *is a GRR* **then**
  return $\mathcal{P}$ as a final component C ;
end

has $n(\mathcal{P})$ notch points. Provided that every node is associated with a location (which is often an inherent requirement of sensing), it is not difficult to identify the boundaries of holes, which make it possible to create a coarse-grained approximate polygon for each of them. There have been numerous solutions proposed in the literature for nodes to discover their locations, and we do not pursue this issue here.

GRR-Decomp recursively divides current polygonal environment into smaller environments (components) until no final environment contains conflicting edges. The process is similar to the convex decomposition in [17]. More specifically, for current polygonal environment $\mathcal{P}$, if there exists a polygon P_i (either the outer polygon or a hole) that contains two conflicting edges, then the algorithm resolves the conflict as follows. The algorithm finds the "most concave" point u on P_i that has the maximum inner angle, and draw a bisector of that angle. The bisector will intersect with some other polygon, say P_j, of $\mathcal{P}$. Assume that the first intersection point is v. If u and v belong to the same polygon P_i (Figure 6(a,b)), then the bisector in effect splits $\mathcal{P}$ into two new polygonal environments $\mathcal{P}_1$ and $\mathcal{P}_2$, and the algorithm will be further performed on $\mathcal{P}_1$ and $\mathcal{P}_2$ separately. If u and v do not belong to the same polygon (Figure 6(c,d)), then the line $\overline{uv}$ will join P_i and P_j to form a new polygon P_{ij}, and $\mathcal{P}$ will be modified accordingly to $\mathcal{P}'$. The algorithm will then be performed on $\mathcal{P}'$. Note that the newly created polygon P_{ij} is no longer a simply polygon, but could be regarded as one by thinking of the new edge $\overline{uv}$ as two different edges $\overline{uv}$ and $\overline{v'u'}$ – this treatment does not affect the algorithm's correctness. For example, in Figure 6(c), the newly created $\mathcal{P}'$ will have a sequence of vertices $u, v, 1, 2, \ldots, 5, v', u', 6, 7$. This recursive process will continue until the current polygonal environment is a GRR, at which point it returns. The algorithm is presented in Algorithm 1.

As an optional optimization, the algorithm can merge neighboring GRRs generated by Algorithm 1,



(a) (b) (c)

Figure 7: Network embedding under GSpring. The nodes are deployed in a non-GRR region (a) and then undergo the GSpring relaxation until reaching an equilibrium. The networks in (b) and (c) have different graph properties: (b) has 541 nodes with average node degree 9.8, and when embedded has a GRR profile polygon, while (c) has 591 nodes with average node degree 4.1, whose profile polygon after embedding is non-GRR.

provided that the combined region is a GRR. Clearly the total number of GRRs generated by Conflict-Resolve($\mathcal{P}$) is no greater than the number of notch points $n(\mathcal{P})$, therefore the merging phase needs at most $n^2(\mathcal{P})$ rounds to finish.

4 Optimization with Graph Embedding

In this section we use graph embedding techniques to further reduce the number of greedily routable components. Graph embedding techniques have been used to assign *virtual coordinates* to nodes in a sensor network in order to enable geographic routing in the absence of physical location information [22], or to improve the performance of greedy routing [3, 15]. We will use an adapted version of the GSpring algorithm, originally proposed in [15], for our problem. This algorithm is an iterative virtual coordinate assignment scheme that simulates the sensor network as a spring mass system, and uses a relaxation algorithm to incrementally increase the convexity of network topology. In its implementation, the algorithm calculates for each node a *region of ownership*, which may cover *conflict* nodes that can attract traffic into local minima when taken as destinations. A node can then "push" those conflict nodes away following several force laws.

Figure 7 demonstrates how GSpring works on a network. Figure 7(a) shows a V-shaped non-GRR sensor field. We create a network in this field by placing sensor nodes according to a certain distribution. Figures 7(b) and 7(c) show the outputs of GSpring for two different node distributions. It can be seen that GSpring can possibly transform a network in a non-GRR region into one with a GRR profile polygon, the outcome depending on node distribution and average node degree. This indicates that under certain conditions, GSpring may reduce the number of components on the basis of GRR-Decomp.

We propose a simple algorithm, termed *EGRR-Decomp*, that produces a set of regions, termed *EGRRs*, by exploiting the embedding ability of a network based on the GRR-Decomp algorithm. Given an output of GRR-Decomp, we perform a further merging routine, termed *Embeddable GRR merging*, which is the same as the merging routine in GRR-Decomp except the condition for two components to be merged. In this merging stage, when checking whether two combined GRRs make a new GRR, the algorithm first fills the combined region with sensor nodes in the same distribution as that of the original network, and then apply a centralized GSpring algorithm, which will produce an embedded graph. Whether the profile polygon of this embedded graph is a GRR then determines whether the two GRRs should be merged.

Since the whole merging process is not meant to find an optimal solution, we simply set a constant number (e.g., 400 in our experiments) of iterative rounds for GSpring. This way the GSpring relaxation can be completed quickly.

Algorithm 2: **DRP forwarding algorithm.**

input : current node u, detination v, and packet P

if $C(v) \neq C(u)$ **then**
  Send P to the next node on the shortest vu-path;
else
  if v *is a neighbor of* u **then**
    Terminate the algorithm;
  else
    Find the node u' that is closest to v among u's neighbors;
    if $|u'v| < |uv|$ **then**
      Send P to u';
    else
      Perform expanding ring search to find a node w such that $|wv| < |uv|$;
      Deliver P to w;
    end
  end
end

5 The Decomposition-based Routing Protocol (DRP)

In this section we design a decomposition-based routing protocol (DRP) based on previous algorithms, and present an analysis of its stretch in a continuous domain.

5.1 The DRP protocol

After running GRR- or EGRR-Decomp, the generated (E)GRR regions, each represented by a sequence of locations and associated with a unique identifier, are broadcasted to the whole network. Every node then learns which region it falls in. A node that does not fall in any region chooses to join a nearest region in terms of Euclidean distance. All the nodes with the same region ID form a component of the network.

During the flooding, a node within a component C also compares with its neighbors within C the distance to C's polygon boundary; if it is the closest one to the boundary, then it marks itself as a *boundary node* of C. The boundary nodes of a component C will be instructed to perform a *joint flooding* operation, which helps every node outside C to establish a shortest path to C. After the component assigning process finishes, the base station does a second round flooding to the network by which it specifies a time t_C for each component C, requiring all C's boundary nodes to start a global flooding simultaneously at the time t. The flooded message carries only the ID of C, and only nodes outside C forward it. The consequence of this procedure is that every node u outside C will be able to receive the flooded message via a shortest path from C. The node u then records the ID of its parent in the shortest path to each of its external components. Note that the boundary nodes may not be perfectly synchronized in time. The impact of time difference can be reduced by increasing the forwarding latency in the flooding, making the multiple floods approximately simultaneous from the receiver's perspective – we only need approximately shortest paths here.

With component IDs assigned and shortest paths to components established, the routing can be done easily. Suppose the source node u in component $C(u)$ wants to route to a destination node v in component $C(v)$. If $C(v) = C(u)$, then an intra-component routing procedure is performed: starting from u, the packet is greedily forwarded to v, the expanding ring subroutine [22] being invoked when local minima occur. If $C(v) \neq C(u)$, then the packet first follows the shortest path to $C(v)$, until reaching the first node in $C(v)$,

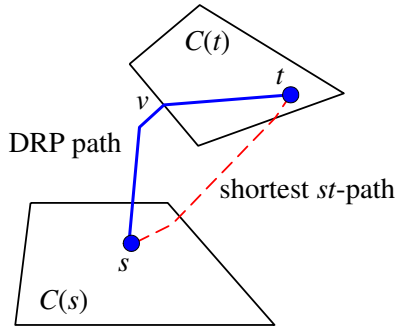


Figure 8: GRR-based DRP has a stretch smaller than 7 in a continuous domain.

at which point it begins intra-component routing. The algorithm is given in Algorithm 2. Note that the expanding ring search is described as an atom operation for clarity.

Depending on whether the components are generated by the GRR-Decomp or EGRR-Decomp algorithms, the DRP protocol is referred to as *GRR-based* or *EGRR-based* DRP, respectively. In the latter case, intra-component routing in C should be performed on the virtual coordinates instead of on real coordinates. The requires an execution of the GSpring protocol [15] after the components assignment is finished.

DRP requires each node to remember $O(n(\mathcal{P}))$ state information, in addition to the state of its local neighborhood.

5.2 Stretch of GRR-based DRP in a continuous domain

Because of the low stretch greedy routing within individual GRRs and the shortest paths between a GRR and its external points, the routing stretch of GRR-based DRP can be bounded by a constant.

Theorem 4. *GRR-based DRP has a stretch smaller than 7 in a continuous domain.*

Proof. Assume two source and destination nodes s and t. If $C(s) = C(t)$, then the stretch is smaller than 3 following Theorem 2. If $C(s) \neq C(t)$, then assume that the shortest path from s to $C(t)$ joins $C(t)$ at the point $v \in C(t)$; see Figure 8. Since the route first goes to v along the shortest path between s and $C(t)$ and then to t greedily, we have $D_{\text{DRP}}(s,t) = D_{\min}(s,v) + D_{\text{DRP}}(v,t) < D_{\min}(s,v) + 3D_{\min}(v,t)$. Using the triangle inequality we get $D_{\min}(v,t) < D_{\min}(s,v) + D_{\min}(s,t)$. Therefore, $D_{\text{DRP}}(s,t) < 4D_{\min}(s,v) + 3D_{\min}(s,t)) < 7D_{\min}(s,t)$, which proves the theorem. □

6 Performance Evaluation

We evaluate the performance of our approach through simulations.

6.1 Stretch of DRP in discrete networks

In this section we show how DRP performs in real networks. Figures 9(a) and (d) show two real maps, one of a residential area and the other of a park, in a city. In the first map, we simulate a network, referred to as STREETS, in which nodes are deployed on the streets. In the second map, we create a network, PARK, by placing nodes on the lawns and several road junctions (so that the network is connected). Sensor nodes are placed on a grid and then perturbed with a random shift following a normal distribution [5, 24]. Every node may have one- or two-hop neighborhood depending on experimental settings. Both uniform disk graph (UDG) and Quasi-UDG models for radio ranges have been considered. We run the GRR-Decomp

(a) Map of several streets. (b) STREETS with 30 GRRs. (c) STREETS with 20 EGRRs.

(d) Map of a park. (e) PARK with 19 GRRs. (f) PARK with 12 EGRRs.

Figure 9: Two network topologies, STREETS and PARK, generated from two real maps. In STREETS, there are 5434 nodes with average node degree 9.5; in PARK, there are 6428 nodes with average node degree 9.9.

and EGRR-Decomp algorithms on both networks to generate a number of components, and then run the DRP protocol for 5,000 randomly picked source-destination pairs. For comparison purposes we have also implemented GPSR [12], a representative localized geographic routing algorithm.

Table 1 presents the greedy success rate (GSR), the proportion of routes found using solely greedy forwarding, and several statistics about *transmission stretch*. Transmission stretch is a variant of route stretch, defined as the ratio of the number of actually transmitted packets during routing between two nodes to the minimum hop count between the same end nodes. First, we can see DRP performs well in all cases, with an average stretch of 1.54 at the highest. In terms of 95th percentile stretch, the worst-case result is 3.53. In contrast, GPSR has an average stretch of 13.2 and 7.13 for STREETS and PARK, respectively, and has a GSR no higher than 20.0% for both networks. This demonstrates how global information is vital for obtaining short paths, and that DSP achieves low stretch routing by using only a small amount of such information (at most 30 table entries in the shown cases). Note that it is not claimed that DRP is superior to localized algorithms like GPSR in all respects, as it is non-local; rather, the emphasis here is that DRP can provide considerable performance gains at only a small price. When such an extra cost is not a serious concern, DRP provides a meaningful choice for low state, low stretch routing in practice.

One can also see that GRR-Decomp generates more components than EGRR-Decomp, but has better GSR and stretches, reflecting a tradeoff between the amount of state information and routing performance.

Table 2 presents the stretch of DRP for Quasi-UDG radio model with parameter $0 \leq \alpha < 1$. Under this model, a link exists between two nodes u and v with probability 1 when $|uv| \leq 1 - \alpha$, and with probability $0 < p < 1$ when $1 - \alpha < |uv| \leq 1 + \alpha$; the link does not exist when $|uv| > 1 + \alpha$. We vary α and adjust p so that the average node degree in different networks remains nearly the same. It can be seen that DRP performs quite well under this model, indicating its robustness to different radio range models.

STREETS					
		GSR	Avg. stretch	5th prtl.	95th prtl.
GRR	1-hop	92.8%	1.07	1.00	1.40
	2-hop	98.4%	1.05	1.00	1.15
EGRR	1-hop	68.2%	1.47	1.00	2.90
	2-hop	98.8%	1.10	1.00	1.29
PARK					
		GSR	Avg. stretch	5th prtl.	95th prtl.
GRR	1-hop	90.0%	1.09	1.00	1.55
	2-hop	99.7%	1.04	1.00	1.17
EGRR	1-hop	69.8%	1.54	1.00	3.53
	2-hop	95.7%	1.26	1.00	1.50

Table 1: Greedy success rate (GSR) and stretch of DRP in STREETS and PARK, under the UDG radio model. Nodes maintain 1- or 2-hop neighborhoods.

STREETS					
		GSR	Avg. stretch	5th prtl.	95th prtl.
GRR	$\alpha = 0.1$	93.7%	1.07	1.00	1.43
	$\alpha = 0.2$	90.7%	1.10	1.00	1.53
	$\alpha = 0.4$	86.0%	1.20	1.00	2.03
PARK					
		GSR	Avg. stretch	5th prtl.	95th prtl.
GRR	$\alpha = 0.1$	90.1%	1.09	1.00	1.54
	$\alpha = 0.2$	93.5%	1.09	1.00	1.35
	$\alpha = 0.4$	86.6%	1.28	1.00	2.28

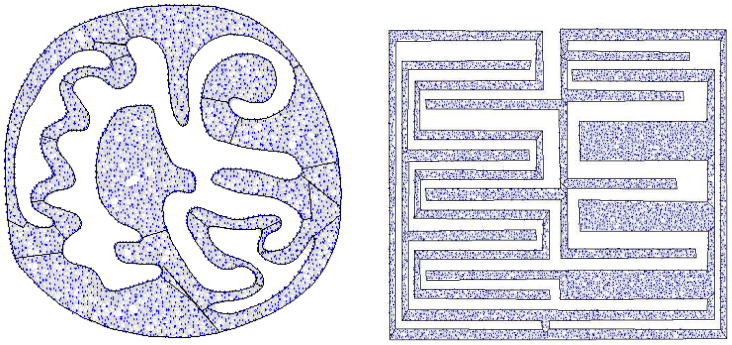
Table 2: Greedy success rate (GSR) and stretch of DRP in STREETS and PARK, under the Quasi-UDG radio model with parameter α. Every node maintains 1-hop neighborhood.

6.2 Message overheads

The main source of message overhead of our routing approach is the joint flooding operations in DRP. When nodes are roughly synchronized, the floods from all the boundary nodes of a component can be viewed as a single flood from a giant single node representing the whole component. So the average message cost per node during this process is very close to the number of components produced by (E)GRR-Decomp, which is in the order of magnitude of tens. Given the quite complex topologies of the examined networks, we believe that the message cost of our design is acceptable to many of today's application scenarios.

6.3 Comparison with alternative decomposition schemes

In addition to local neighborhood state, every node needs to maintain as many (inter-component) routing table entries as there are components in the network. We examine the state information required by DRP as compared with the convex decomposition approach in [23]. In [23], the sensor network is divided into convex partitions and a landmark node is picked in each partition; the routing is done following a similar intra- and inter-partition paradigm to DRP. It is shown that such an approach substantially reduces the per-node state as compared with some previous schemes. Here we show that compared with convex decomposition, the GRR approach can provide a further significant reduction in state overheads, with little sacrifice on routing performance.



(a) FISHEYE. (b) MAZE.

Figure 10: Benchmark geometric shapes for decomposition, FISHEYE and MAZE. The pictures show the results of GRR decomposition.

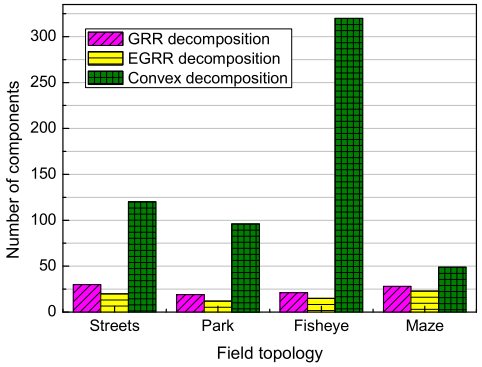


Figure 11: Number of components produced by different decomposition schemes.

The convex decomposition algorithm we use for comparison is the ACD algorithm proposed in [17] (which generates fewer components than the protocol in [23] since ACD is centralized). Figure 10 shows the number of GRR components, EGRR components, and convex components produced by ACD for four field topologies. Apart from the STREETS and PARK topologies, two additional benchmark examples are taken from [17].

The results in Figure 11 indicate that our decomposition algorithm divides the field into much fewer components than ACD does.[2] To see how the routing performance is affected, we also implemented DRP based on the ACD results. For example, in STREETS, the average and 95th percentile stretch are 1.04 and 1.10, respectively, with a GSR of 97.6%. In the same setting, the GRR approach has performance only 5% lower than that of ACD (see the first line of Table 1), with only a quarter of ACD's cost in terms of number of components (30 vs. 120). The same observation is made for other test cases. This suggests that our (E)GRR-based routing protocol needs much lower state to achieve comparable routing performance.

[2] ACD generates a high number of components for the FISHEYE topology because the original topology contains many short line segments on boundary.

7 Related work

The best known geographic routing algorithm, GPSR [12], is shown to achieve a stretch close to 1 in a regular and uniform network, but in a topologically complex network could have a stretch as high as $\Omega(c^2)$ [13], where c is the shortest path length between a node pair under consideration. Kuhn et al. [13] improve this to $\Omega(c)$, which is optimal for localized algorithms. The VIGOR algorithm [24] achieves $\Theta(1)$ stretch, with extra non-local state proportional to the complexity of large topological features of the network (e.g., number of big holes). Other representative work utilizing non-local network topological information include [5, 8]. To enable geographic routing without using physical location information, graph embedding techniques have been successfully used to assign virtual coordinates [20, 22, 25] to network nodes. It is shown that the resulting stretch is comparable with or even better than using real coordinates in some typical circumstances.

We were not the first to consider partition-based routing in sensor networks. In [8], the authors propose to divide the sensor field into tiles where local greedy routing tends to work well. The protocol involves a global preprocessing stage by which each node is provided a global map of titles. The routing is first planned using this map and then realized within each title using local neighborhood information. In [29], the network is partitioned into pieces that have nice shapes, which are considered beneficial to a number of applications including routing. Observing that the convexity of a network component allows highly efficient greedy routing, Tan et al. [23] consider partitioning a sensor network into a set of convex components. In this paper, we show that a component does not need to be convex for efficient greedy routing, thus a network can be divided into much fewer components while enjoying low-stretch greedy routing.

Small state routing tries to minimize nodes' routing table size while providing close-to-optimal route lengths. In [19], Mao et al. propose a small state and small stretch routing protocol for large wireless sensor networks. This problem is well known as *compact routing* in theoretic fields and a large body of literature can be found on it; see [11] for an overview. For a general n-node graph, there have been routing schemes with stretch $\approx k$ that require $\approx O(n^{1/k})$ bits per node, which are asymptotically tight up to poly-logarithmic factors. Better results have been obtained on restricted classes of graphs such as low doubling dimension graphs [1], unit disk graphs [9], etc. Sharing a similar goal as previous work, our work takes a different approach with the implicit assumption is that local network connectivity exhibits a certain degree of uniformity – a common scenario found in reality. This allows us to concentrate on the significant geometric characteristics of the field, which dominate the performance of greedy routing.

Polygon decomposition has many theoretical and practical applications. A typical problem is to divide a polygon into a minimum number of convex components. While the solution of such a problem for simple polygons without holes can be found in polynomial time [6], the problem is NP-hard for polygons containing holes [18, 21].

8 Conclusion

We have studied the problem of decomposing a network into a minimum set of components where greedy geographic routing performs well. We explore its hardness and develop its solutions. We believe that the proposed algorithms provide a natural solution to networks with complex topologies when we pursue low-state and low-stretch routing.

Regarding the fundamental nature of the minimum decomposition problem, our results reveal only part of the picture. For example, our NP-hardness result holds precisely for a polygonal environment containing holes, but it is unclear whether this remains true for more restricted non-hole cases. Currently, the definitions and techniques developed for the Min-GRR problem are limited to only 2D sensor networks, it might be interesting to consider a similar decomposition problem for 3D cases.

References

[1] I. Abraham, C. Gavoille, A. Goldberg, and D. Malkhi. Routing in Networks with Low Doubling Dimension. In *Proc. of ICDCS 2006*

[2] H. Alt and E. Welzl. Visibility graphs and obstacle-avoiding shortest paths. *Zor-Zeitschrift fur Operation Research* 32:145-164, 1988.

[3] N. Arad and Y. Shavitt. Minimizing recovery state in geographic ad-hoc routing. *Proc. of MobiHoc 2006*.

[4] X. Bai, S. Kumary, D. Xuan, Z. Yun, and T. H. Lai. Deploying Wireless Sensors to Achieve Both Coverage and Connectivity. *Proc. of MobiHoc'06*. Italy, 2006.

[5] J. Bruck, J. Gao, A. Jiang. MAP: Medial Axis Based Geometric Routing in Sensor Networks, *Proc. of MobiCom 2005*.

[6] B. Chazelle and D. P. Dobkin. Decomposing a polygon into its convex parts. *Proc. ACM STOC 1979*.

[7] S. M. Das, H, Pucha, and Y. C. Hu. Performance Comparison of Scalable Location Services for Geographic Ad Hoc Routing. In *INFOCOM 2005*.

[8] Q. Fang, J. Gao, L. Guibas, V. de Silva, and L. Zhang. GLIDER: Gradient landmark-based distributed routing for sensor networks. *Proc. INFOCOM 2005*.

[9] R. Flury, S. V. Pemmaraju, and R.Wattenhofer. Greedy Routing with Bounded Stretch. *Proc. INFOCOM 2009*.

[10] R. Fonseca, S. Ratnasamy, J. Zhao, C. T. Ee, D. Culler, S. Shenker, and I. Stoica. Beacon vector routing: Scalable point-to-point routing in wireless sensornets. *Proc. of NSDI 2005*.

[11] C. Gavoille. Routing in distributed networks: overview and open problems. *ACM SIGACT News – Distributed computing column*, 32(1):36-52, 2001

[12] B. Karp and H. T. Kung. GPSR: Greedy perimeter stateless routing for wireless networks. In *Proc. of MobiCom 2000*.

[13] F. Kuhn, R. Wattenhofer, and A. Zollinger. An Asymptotically Optimal Geometric Mobile Ad Hoc Routing. In *Proc. of ACM DIALM-POMC Joint Workshop on Foundations of Mobile Computing*, 2002.

[14] D. Lichtenstein. Planar Formulae and Their Uses. *SIAM Journal on Computing*, vol. 11, pp. 329-343, 1982.

[15] B. Leong, B. Liskov, and R. Morris. Greedy virtual coordinates for geographic routing. In *ICNP 2007*.

[16] M. Li and Y. Liu. Rendered Path: Range-Free Localization in Anisotropic Sensor Networks with Holes. In *Proc. of MobiCom 2007*.

[17] J-M Lien and N. M. Amato. Approximate Convex Decomposition of Polygons. In *Proc. ACM Symp. Computational Geometry 2004.*

[18] A. Lingas. The power of non-rectilinear holes. In *Proc. 9th Internat. Colloq. Automata Lang. Program* (ICALP), volume 140, LNCS 369-383. 1982.

[19] Y. Mao, F. Wang, L. Qiu, S. S. Lam, and J. M. Smith. S4: Small state and small stretch routing protocol for large wireless sensor networks. In *Proc. of NSDI 2007*.

[20] J. Newsome and D. Song. Gem: Graph embedding for routing and data-centric storage in sensor networks without geographic information. In *Proc. of SenSys 2003*.

[21] J. O'Rourke and K. J. Supowit. Some NP-Hard Polygon Decomposition Problems. *IEEE Transactions on Information Theory*. 29(2), March 1983.

[22] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, and I. Stoica. Geographic routing without location information. In *MobiCom 2003*.

[23] G. Tan, M. Bertier and A-M. Kermarrec. Convex Partition of Sensor Networks and Its Use in Virtual Coordinate Geographic Routing. *Proc. of INFOCOM 2009*.

[24] G. Tan, M. Bertier and A-M. Kermarrec. Visibility-Graph-based Shortest-Path Geographic Routing in Sensor Networks. *Proc. of INFOCOM 2009*.

[25] M. Tsai, H. Yang, and W. Huang. Axis-Based Virtual Coordinate Assignment Protocol and Delivery-Guaranteed Routing Protocol in Wireless Sensor Networks. *Proc. of INFOCOM 2008*.

[26] X. Wang, G. Xing, Y. Zhang, C. Lu, R. Pless and C. Gill. Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks. *Proc. ACM SenSys 2003*.

[27] Y. Xu, J. Heidemann, D. Estrin. Geography informed Energy Conservation for Ad Hoc Routing. *Proc. of MobiCom 2001*.

[28] J. Wu and S. Yang. SMART: A Scan-Based Movement-Assisted Sensor Deployment Method in Wireless Sensor Networks. *Proc. of INFOCOM 2005*.

[29] X. Zhu, R. Sarkar, and J. Gao. Shape segmentation and applications in sensor networks. In *Proc. of INFOCOM 2008*.

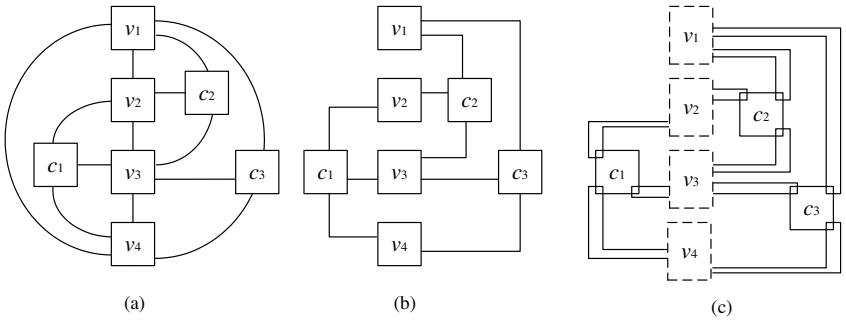


Figure 12: The graph of the P-3SAT Boolean formula $B = \{c_1, c_2, c_3\}$, where $c_1 = (\overline{v_2}+v_3+\overline{v_4}), c_2 = (\overline{v_1}+v_2+v_3)$, and $c_3 = (v_1 + v_3 + v_4)$.

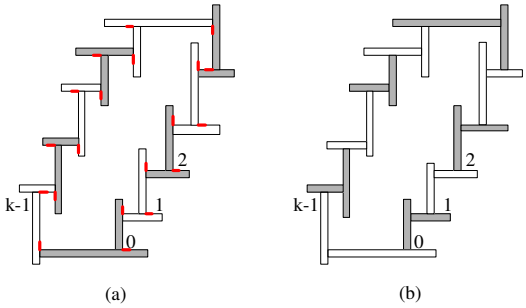


Figure 13: Basic variable loops. (a) A possible TRUE decomposition; (b) a possible FALSE decomposition.

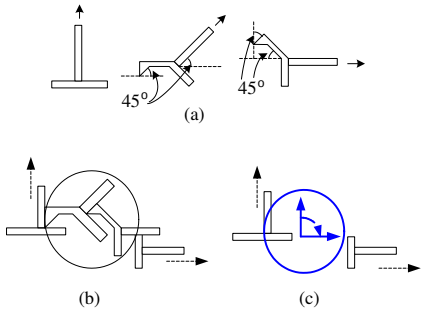


Figure 14: Bent $\perp$ blocks and bend sets.

A Proof of Theorem 3

Let B be a 3SAT Boolean formula $B = \{c_1, c_2, \ldots, c_m\}$, where each $c_i (1 \leq i \leq m)$ is a disjunctive 3-literal clause over the Boolean variable set $\{v_1, v_2, \ldots, v_n\}$. Let $G(B)$ be the graph corresponding to B with vertex set $\{v_1, \ldots, v_n\} \cup \{c_1, \ldots, c_m\}$, and edge set $E_1 \cup E_2$, where $E_1 = \{(v_i, c_j) : v_i \in c_j \vee \overline{v_i} \in c_j\}$ and $E_2 = \{(v_j, v_{j+1}) : 1 \leq j < n\} \cup \{(v_n, v_1)\}$. B is a P-3SAT Boolean formula if $G(B)$ is planar. Figure 12(a) shows an example of the P-3SAT Boolean formula $B = \{c_1, c_2, c_3\}$. The P-3SAT problem asks: Given a P-3SAT Boolean formula B, does there exist a truth assignment for the Boolean variables such that all the clauses in B are satisfied simultaneously?

We shall use a modified version of P-3SAT, called *rectilinear P-3SAT* (RP-3SAT), for showing the hardness of the problem. In RP-3SAT, the edges are all rectilinear, and every link makes at most one turn. Moreover, the edges in E_2 will not be used. We identify three types of links between variables and clauses: straight link, down-turn link, and up-turn link. Figure 12(b) shows the rectilinear version of the graph in Figure 12(a). From the NP-completeness of P-3SAT, it is possible to verify that RP-3SAT is NP-complete as well.

A.1 Constructing truth-setting components

First we describe the construction of a truth-setting component, the structure that corresponds to a Boolean variable in an RP-3SAT instance. Such a structure, termed a *variable loop*, is composed of a cycle of k connected $\perp$-shaped blocks (see Figure 13), denoted by $\perp_0, \perp_1, \ldots, \perp_{k-1}$, in an anti-clockwise direction. Each $\perp_i$ has a *horizontal bar* and a *vertical bar*, denoted by $\perp_i^h$ and $\perp_i^v$, respectively. A bar can extend freely on its open ends when needed, and has an arbitrarily small width. A variable loop can also be represented by

a cycle of bars, $\perp_0^h, \perp_0^v, \perp_1^h, \perp_1^v, \ldots, \perp_{k-1}^h, \perp_{k-1}^v$. The open direction of the vertical bar of $\perp_i$ determines $\perp_i$'s *direction*, denoted by $D(\perp_i)$. By default, a $\perp$ block's direction is $\frac{\pi}{2}$. Assuming one is facing the direction of $\perp_i$, then the block $\perp_{i+1}$ (hereinafter $i+1$ is always taken modulo k if not stated otherwise) can either be on the right or left side of $\perp_i$. In the construction of a variable loop, two $\perp$ block placement rules must be followed: (1) $\perp_{i+1}^h$ is connected to $\perp_i^v$; (2) $\perp_{i+1}$ is at the right side of $\perp_i$ when $D(\perp_{i+1}) = D(\perp)$, or at the left side of $\perp_i$ when $D(\perp_{i+1}) = D(\perp_i) + \pi/2$ (that is, $\perp_{i+1}$ has a reverse direction of $\perp_i$). Figure 13 shows a *basic variable loop*, in which a $\perp$ block has a direction of either $\frac{\pi}{2}$ or $\frac{3\pi}{2}$.

In a basic variable loop, every block $\perp_i$ has two *critical segments*, s_i^0 and s_i^1, which are two arbitrarily short line segments on its boundary (see the thick lines in Figure 13). s_i^0 is on the boundary of $\perp_i^h$, and is adjacent to $\perp_i^v$ on the right (resp. left) if $\perp_{i+1}$ is on the right (resp. left) of $\perp_i$; s_i^1 is on the boundary of $\perp_i^v$, and is on the top (resp. beneath) $\perp_{i+1}^h$ if $\perp_{i+1}$ has the same (resp. different) direction as $\perp_i$.

Definition 3. **TRUE and FALSE GRR Decomposition**. *A GRR decomposition of a basic variable loop is said to be* TRUE *if every pair of s_i^0 and s_i^1 belongs to the same GRR, and is said to be* FALSE *if every pair of s_i^1 and s_{i+1}^0 belongs to the same GRR.*

Figures 13(a) and (b) provide two examples of TRUE and FALSE decompositions of a basic variable loop, shown in alternating grey/while colors. Note that a TRUE or FALSE decomposition is not unique.

Lemma 1. *A basic variable loop of k $\perp$ blocks can be minimally decomposed into k GRRs, and such a decomposition must be either* TRUE *or* FALSE.

Proof. Let the critical segments on the variable loop be $s_0^0, s_0^1, \ldots, s_{k-1}^0, s_{k-1}^1$. It is easy to see that any GRR can include at most two critical segments, otherwise the GRR will contain conflicting edges which violates the property of a GRR. Since there are $2k$ critical segments, the variable loop will contain at least k GRRs. It is also possible to verify that any two critical segments included in a GRR must be consecutive; therefore a minimum GRR decomposition must be either TRUE or FALSE. □

The bars of a $\perp$ block may be bent and slightly deformed, forming the shape shown in Figure 14(a). Such operation results in the change of a $\perp$ block's direction, as indicated by the arrows near the $\perp$ blocks. Two bent $\perp$ blocks put together can change the direction of succeeding $\perp$ blocks to 0 or π. Such two $\perp$ blocks are called a *bend set*. Figure 14(b) shows an example of a bend set enclosed in a circle, which is simplified to a dial-plate-like symbol in Figure 14(c).

A basic variable loop can be extended to a *full variable loop* by drawing "arms" out of its right or left side. Arms simulate the links between the Boolean variables and the clauses in $G(B)$. An arm may or may not contain bends, and conforms to the two $\perp$ block placement rules as well. Figure 15 provides an example of a full variable loop with a single arm containing two bend sets at its right side. An arm must contain an even number of bend sets in order to restore the direction of $\perp$ blocks to $\frac{\pi}{2}$ or $\frac{3\pi}{2}$ when it returns to the basic variable loop.

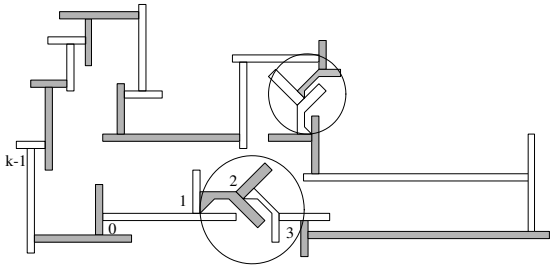


Figure 15: A full variable loop with two bend sets.

The definition of critical segments and TRUE/FALSE GRR decomposition can be easily extended to full variable loops.

Lemma 2. *A full variable loop of k $\perp$ blocks, either containing bend sets or not, can be minimally decomposed into k GRRs, and such a decomposition must be either* TRUE *or* FALSE.

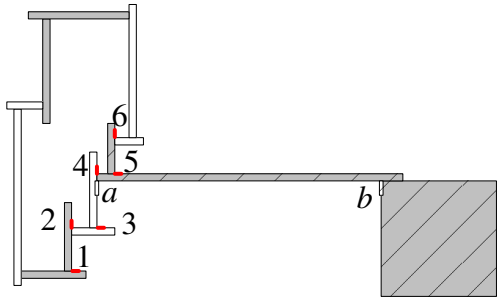


Figure 16: A variable loop connected to a clause junction via a straight link and appearing in the clause as positive.

A.2 Clause junctions and links to truth-setting components

We are now ready to describe the construction of clause junctions that correspond to disjunctive clauses in an RP-3SAT instance, and how they are connected to variable loops. A clause junction is simply a square area (see Figure 17). Out of its four corners, three are selected to be the *ports*, where the clause junction is connected to its associated variable loops through arms. Recall that in the RP-3SAT problem, a variable connects to a clause via a straight, down-turn, or up-turn link. When the clause junction is at the right side of the variable loop[3], an arm connects the clause junction to a variable loop via one of three ports, namely the *top-left port*, the *top-right port*, and the *bottom-right port*, depending on the type of the link: (1) if the link is straight, then the arm joins the clause junction via the top-left port; (2) if the link is down-turn, then the arm joins the clause junction via the top-right port; (3) if the link is up-turn, then the arm joins the clause junction via the bottom-right port.

The structure of an arm differs also depending on whether the variable v appears in a clause c as positive or negative. Thus there are six cases for the construction of an arm; see Figure 17 for an illustration. The key consideration in constructing an arm is that the arm should be able to integrate the clause square c into the variable loop's area "for free" (i.e., without increasing the number of GRRs in the combined area) if and only if a TRUE/FALSE decomposition, where the value taken satisfies c, is applied to the variable loop.

Lemma 3. *Following the scheme in Figure 17, the clause junction c can be integrated into the variable loop v's area without increasing the minimum number of GRRs if and only if a* TRUE/FALSE *decomposition, where the value taken satisfies c, is applied to the variable loop.*

Proof. First consider the case shown in Figure 17(a) where the variable loop connects to the clause junction via a straight link and the variable appears in the clause as positive. Figure 16 provides a closer look of this scenario. In the figure, we identify two additional critical segments, a and b. When a TRUE decomposition is applied to the variable loop, as shown by the alternating while/grey colors, the clause square can be incorporated into the $\perp$ block that contains the critical segments 5 and 6, without introducing conflicting edges. That is, the combined (shaded grey) area of the $\perp$ block and the clause square can make a single GRR. Therefore, under the TRUE decomposition, the addition of the clause square does not increase the minimum number of GRRs of the variable loop. Now consider a FALSE decomposition of the variable loop instead. If the minimum number of GRRs of the variable loop were not to be increased with the addition of the clause square, then the critical segments a and b must share the same component that includes the critical segments 4 and 5. However, these four segments cannot co-exist in a single GRR because a conflicts with b. This means that under the FALSE decomposition, the clause square cannot be incorporated by the variable loop for free.

[3]Due to the symmetry of the right and left sides of a variable loop, we only need to consider the right side.

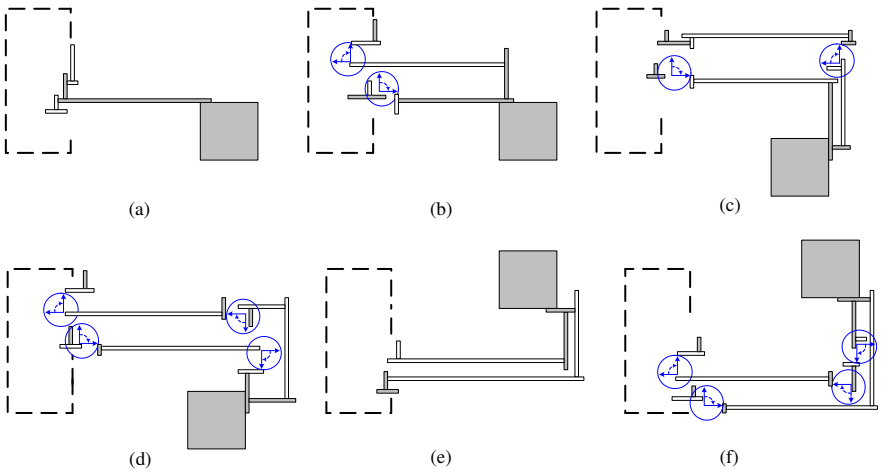


Figure 17: The six cases of a variable loop v connected to a clause junction c. (a) v connects to c via a straight link and appears in c as positive; (b) v connects to c via a straight link and appears in c as negative; (c) v connects to c via a down-turn link and appears in c as positive; (d) v connects to c via a down-turn link and appears in c as negative; (e) v connects to c via an up-turn link and appears in c as positive; (f) v connects to c via an up-turn link and appears in c as negative.

The same argument can be applied to the other five cases in Figures 17(b)-(f). The details are omitted. There results prove the lemma. □

A.3 Constructing a polygonal region for an RP-3SAT instance

Given an RP-3SAT instance B of n Boolean variables and m clauses, the constructed polygonal region consists of n variable loops, each corresponding to a Boolean variable, and m clause squares, each corresponding to a clause. The positions of these components reflect the relative positions of the variables and clauses in $G(B)$. Arms of a variable loop are brought from a variable loop to the clauses that it participate as illustrated previously. Figure 12(c) schematically shows the constructed polygonal region for the RP-3SAT example shown in Figure 12(a); the full picture of the construction is given in Figure 18.

There is no limit to the length of a bar in a $\perp$ block, so the magnitude of the variable loop's size on the plane is unimportant. The $\perp$ block's ability to extend freely makes it possible to construct an arm with at most a constant number of $\perp$ blocks, and consequently, to construct a full variable loop with $O(m)$ $\perp$ blocks. Since there are n full variable loops and m clauses, the construction of a complete polygonal region can be done within $O(mn)$ time. We thus have the following lemma.

Lemma 4. *The construction of a polygonal region for an RP-3SAT instance requires only polynomial time.*

Given an RP-3SAT instance, we construct a polygonal region using the introduced approach in polynomial time. Suppose the total number of $\perp$ blocks used is K, then according to the minimum decomposition property of variable loops, the overall polygonal region can be decomposed into K GRRs if and only if all the m clause junctions can be integrated into the variable loops for free, which, according to Lemma 3, means that all the clauses in the RP-3SAT instance are satisfied. This proves the NP-hardness of the problem.

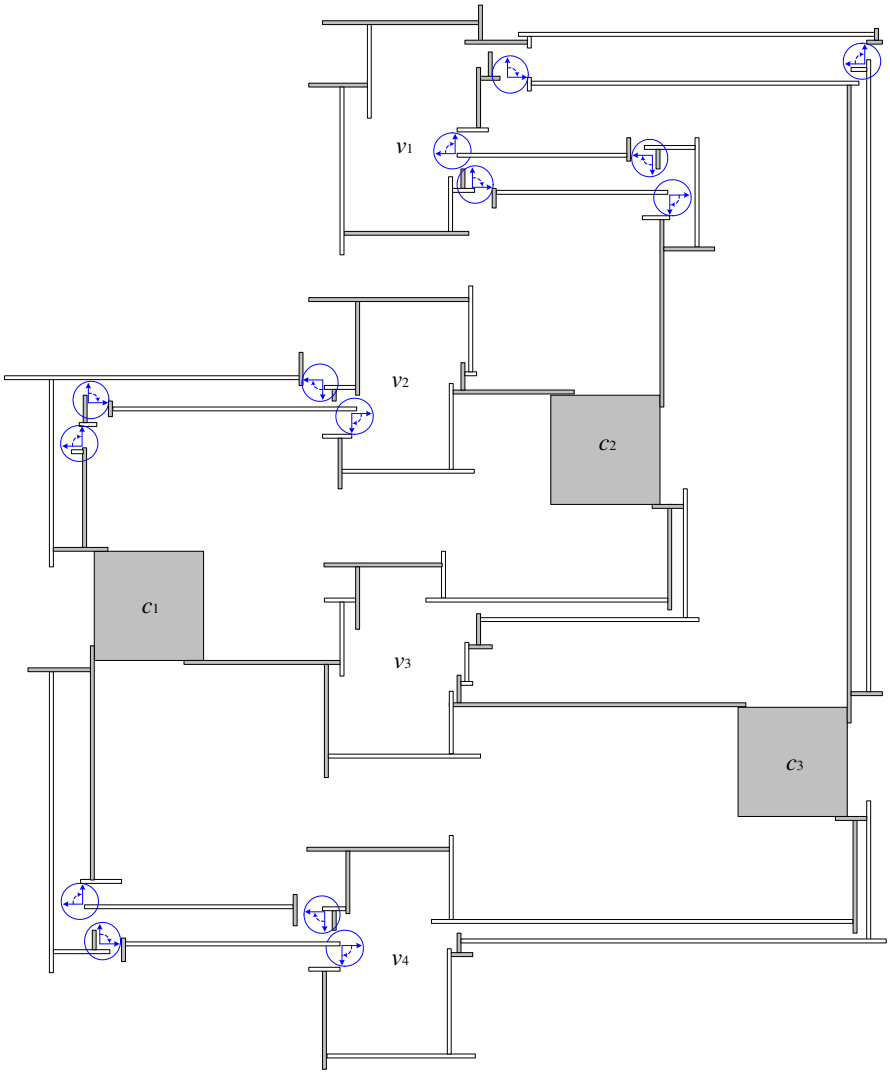


Figure 18: Constructed polygon region for Boolean formula $B = \{c_1, c_2, c_3\}$, where $c_1 = (\overline{v_2} + v_3 + \overline{v_4}), c_2 = (\overline{v_1} + v_2 + v_3)$, and $c_3 = (v_1 + v_3 + v_4)$. The region consists of p regular $\perp$ blocks, q bend sets, and r clause junction squares. It can be minimally decomposed into $p + 2q$ GRRs if and only if there exists a truth assignment for the variables v_1, v_2, v_3, v_4 such that all the clauses in B can be satisfied.